This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1 -5. (Cancelled)

6. (Previously presented) The method of claim 31 further comprising providing a cursor on any type of query executed.

7. (Previously presented) The method of claim 31 wherein a programming model for an out-of-process application is symmetrical with an in-process programming model for the DBMS.

8. (Previously presented) The method of claim 31 further comprising the marshaling of data between an unmanaged layer and a managed layer.

9. (Previously presented) The method of claim 31 wherein an application operation from a group of operations comprising functions, procedures, and triggers is executed directly in the DBMS.

10. (Previously presented) The method of claim 9 wherein a result is returned by the DBMS to a client based on the execution of the application operation by the DBMS.

11-15. (Cancelled)

16. (Previously presented) The system of claim 37 further comprising a subsystem for providing a cursor on any type of query executed.

17. (Previously presented) The system of claim 37 wherein a programming model for an out-of-process application is symmetrical with an in-process programming model for the DBMS.

18. (Previously presented) The system of claim 37 further comprising a subsystem for the marshaling of data between an unmanaged layer and a managed layer.

19. (Previously presented) The system of claim 37 further comprising a subsystem for an application operation from a group of operations comprising functions, procedures, and triggers to be executed directly in the DBMS.

20. (Previously presented) The system of claim 19 further comprising a subsystem by which a result is returned by the DBMS to a client based on the execution of the application operation by the DBMS.

21-25. (Canceled)

26. (Previously presented) The computer-readable instructions of claim 43 further comprising instructions for providing a cursor on any type of query executed.

27. (Previously presented) The computer-readable instructions of claim 43 further comprising instructions for a programming model for an out-of-process application that is symmetrical with an in-process programming model for the DBMS.

28. (Previously presented) The computer-readable instructions of claim 43 further comprising instructions for the marshaling of data between an unmanaged layer and a managed layer.

29. (Previously presented) The computer-readable instructions of claim 43 further comprising instructions for an application operation from a group of operations comprising functions, procedures, and triggers to be executed directly in the DBMS.

30. (Previously presented) The computer-readable instructions of claim 29 further comprising instructions whereby a result is returned by the DBMS to a client based on the execution of the application operation by the DBMS.

31. (Currently amended) A computer-implemented method for executing .NET managed code in a database management system (DBMS) having a database server, the method comprising:

executing instructions from a memory in the database server invoking .NET managed code;

invoking an invocation context in the database server, wherein the invocation context ~~comprises~~ is based on at least a context class, wherein the context class includes information comprising a ~~providing access to~~ a ~~client's~~ connection context of a client, a ~~client's~~ command context of the client, a ~~client's~~ transaction context of the client, a ~~client's~~ pipe context of the client, and a ~~client's~~ trigger context of the client;

separating the .NET managed code into an immutable part and a mutable part;

exposing the ~~client's connection~~ context class to the database server through the utilization of an in-process provider, wherein the in-process provider keeps track of unmanaged data that is referenced from a managed space and prevents access of the unmanaged data outside a managed execution frame;

executing the .NET managed code in the database server based on the invocation context and the separation into immutable and mutable parts~~, wherein the code is executed under the client's connection context~~; and

storing information for the ~~client's connection~~ context class in said memory.

32. (Canceled)

33. (Previously presented) The method of claim 31, further comprising a client, wherein the client is a .NET application and the in-process provider is an ADO.net in-process provider.

34. (Canceled)

35. (Currently amended) The method of claim 31, wherein invoking .NET managed code in the database server as a result of the invocation context~~a client trigger~~.

36. (Currently amended) The method of claim 31, wherein the in-process provider supports more than one pending executing command for ~~a client~~ a connection of the client.

37. (Currently amended) A system for executing application code in a database management system (DBMS) comprising a processor and a memory, the system comprising:

at least one processor comprising:

a subsystem for invoking .NET managed code and an invocation context in the database server, wherein the invocation context ~~comprises~~ is based on at least a context class, wherein the context class contains information comprising a ~~providing access to~~ a ~~client's~~ connection context of a client, a ~~client's~~ command context of the client, a ~~client's~~ transaction context of the client, a ~~client's~~ pipe context of the client, and a ~~client's~~ trigger context of the client;

a subsystem for separating the .NET managed code into an immutable part and a mutable part;

a subsystem for exposing the ~~client's connection~~ context class to the database server through the utilization of an in-process provider, wherein the in-process provider keeps track of unmanaged data that is referenced from a managed space and prevents access of the unmanaged data outside a managed execution frame;

a subsystem for executing the .NET managed code in the database server based on the invocation context and the separation into immutable and mutable parts, wherein the code is executed under the client's connection context; and

a computing memory communicatively coupled to the processor, the computing memory operable to store information for the client's connection context.

38. (Currently amended) The system of claim 37, wherein the invocation context further comprises:

a command with a state execution context;

[[a]]the transaction context of a client associated with a command;

a path through which requests and results may be sent or received between the client and database server; and

a forward-only cursor on top of statement execution results.

39. (Previously presented) The system of claim 37, further comprising a client subsystem,

wherein the client subsystem comprises a .NET application, and

wherein the in-process provider is an ADO.net in-process provider.

40. (Canceled)

41. (Currently amended) The system of claim 37, wherein invoking. NET managed code in the database server is a result of the invocation context~~a client trigger~~.

42. (Currently amended) The system of claim 37, wherein the in-process provider supports for more than one pending executing command for ~~a client~~ a connection of the client.

43. (Currently amended) A computer-readable storage medium comprising computer-readable instructions for executing application code in a database management system (DBMS), the computer-readable instructions comprising instructions for:

receiving application code, rewritten as .NET managed code, from an application;

invoking .NET managed code and an invocation context in the database server, wherein the invocation context ~~provides access to a client's connection context~~ is based on at least a context class, wherein the context class contains information comprising a connection context of a client, a command context of the client, a transaction context of the client, a pipe context of the client, and a trigger context of the client;

separating the .NET managed code into an immutable part and a mutable;

exposing the ~~client's connection~~ context class to the database server through the utilization of an in-process provider, wherein the in-process provider keeps track of

unmanaged data that is referenced from a managed space and prevents access of the unmanaged data outside a managed execution frame; and

executing the .NET managed code in the database server based on the invocation context and the separation into immutable and mutable parts.

44. (Previously Presented) The computer-readable instructions of claim 43, wherein exposing the invocation context further comprises exposing at least one of:

a command with a state execution context;

a transaction context associated with a command;

a path through which requests and results may be sent or received between the client and database server;

a trigger context, wherein the trigger results from an operation of the client; or

a forward-only cursor on top of statement execution results.

45. (Previously presented) The computer-readable instructions of claim 43, further comprising a client,

wherein the client comprises a .NET application, and

wherein the in-process provider is an ADO.net in-process provider.

46. (Canceled)

47. (Currently amended) The computer-readable instructions of claim 43, wherein invoking. NET managed code in the database server is a result of the invocation context~~a client trigger~~.

48. (Currently amended) The computer-readable instructions of claim 43, wherein the in-process provider supports more than one pending executing command for ~~a client~~a connection of the client.